	Type	L #	Hits	earch Text	DBs	Time Stamp	Comments
1	BRS	L1	156	insert$ with data near2 byte and 'data word'	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:41	
2	BRS	L2	0	1 and cycle and immediate adj word	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:41	
3	BRS	L3	107	1 and cycle	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:41	
4	BRS	L4	40	3 and alignment$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:52	
5	BRS	L5	2	4 and preceding adj cycle	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:43	
6	BRS	L6	2	4 and preceding near3 cycle	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:43	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
7	BRS	L7	40	4 and select$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:52	
8	BRS	L8	37	7 and shift$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	
9	BRS	L9	37	8 and register and control	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	
10	BRS	L10	37	9 and point$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:54	
11	BRS	L11	23	10 and mask	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:45	
12	BRS	L12	28	10 and mask$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:54	

	Type	L #	Hits	earch T xt	DBs	Time tamp	Comments
13	BRS	L13	1	10 and insert$5 near3 point$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:47	
14	BRS	L14	37	10 and insert$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:55	
15	BRS	L15	28	12 and insert$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:48	
16	BRS	L16	1	15 and re-align$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:48	
17	BRS	L17	14	15 and concatenat$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:55	
18	IS&R	L18	302	(712/300).CCLS.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:51	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
19	IS&R	L19	0	("17and18").PN.	US-PGPUB; USPAT; EP ; JPO; DERWENT; IBM_TDB	2004/12/10 11:51	
20	IS&R	L20	0	("18andinsert$5").PN.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:51	
21	BRS	L21	34	18 and data near3 insert$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:52	
22	BRS	L22	22	21 and alignment$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:52	
23	BRS	L23	31	21 and select$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	
2	BRS	L24	22	22 and select$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
25	BRS	L25	28	23 and shift$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	
26	BRS	L26	28	25 and shift$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	
27	BRS	L27	23	26 and register and control	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:53	
28	BRS	L28	20	27 and point$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:54	
29	BRS	L29	17	28 and mask$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:54	
30	BRS	L30	17	29 and point$3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:54	

	Type	L #	Hits	Search Text	DBs	Time Stamp	Comments
31	BRS	L31	17	30 and insert$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:55	
32	BRS	L32	4	31 and concatenat$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	2004/12/10 11:55	

PORTAL
US Patent & Trademark Office

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: [✓ The ACM Digital Library] The Guide

data and insertion and insert and cycle and register and selec

THE ACM DIGITAL LIBRARY

Feedback Report a problem Satisfacti

Terms used **data** and **insertion** and **insert** and **cycle** and **register** and **select** and **byte** and **data word** and **control** and **alignment** and **intermediate** and **pointer** and **operator** and **logic** and **cancatenate** and **current** and **p**

Sort results by relevance

Display results expanded form

Save results to a Binder

Search Tips

Open results in a new window

Try an Advanced Search

Try this search in The ACM

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

Best 200 shown

1 Data and memory optimization techniques for embedded systems

P. R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandercappelle, P. G. Kjeldsb

April 2001 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 6 Issue 2

Full text available: pdf(339.91 KB) Additional Information: full citation, abstract, references, citings, index terms

We present a survey of the state-of-the-art techniques used in performing data and memory-related optimiza systems. The optimizations are targeted directly or indirectly at the memory subsystem, and impact one or m important cost metrics: area, performance, and power dissipation of the resulting implementation. We first ex independent optimizations in the form of code transoformations. We next cover a broad spectrum of optimiza

Keywords: DRAM, SRAM, address generation, allocation, architecture exploration, code transformation, data optimization, high-level synthesis, memory architecture customization, memory power dissipation, register fil survey

2 Compiler transformations for high-performance computing

David F. Bacon, Susan L. Graham, Oliver J. Sharp

December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4

Full text available: pdf(6.32 MB) Additional Information: full citation, abstract, references, citings, index terms, re

In the last three decades a large number of compiler transformations for optimizing programs have been impl optimizations for uniprocessors reduce the number of instructions executed by the program using transformat analysis of scalar quantities and data-flow techniques. In contrast, optimizations for high-performance supers parallel processors maximize parallelism and memory locality with transformations that rely on tracking the p

Keywords: compilation, dependence analysis, locality, multiprocessors, optimization, parallelism, superscala vectorization

3 The evolution of the Sperry Univac 1100 series: a history, analysis, and projection

B. R. Borgerson, M. L. Hanson, P. A. Hartley

January 1978 **Communications of the ACM**, Volume 21 Issue 1

Full text available: pdf(1.89 MB) Additional Information: full citation, abstract, citings, index terms

The 1100 series systems are Sperry Univac's large-scale mainframe computer systems. Beginning with the 11 series has progressed through a succession of eight compatible computer models to the latest system, the 11 1977. The 1100 series hardware architecture Is based on a 36-bit word, ones complement structure which ob from storage and one from a high-speed register, or two operands from high-speed registers. The 1100 Oper

Keyw rds: 1100 computer series, computer architecture, data management systems, end user facilities, exe software, multiprocessing, multiprogramming, operating system, programming languages

4 Computing curricula 2001
September 2001 **Journal on Educational Resources in Computing (JERIC)**
Full text available: pdf(613.63 KB) html(2.78 KB) Additional Information: full citation, references, citings, index terms

5 Fast detection of communication patterns in distributed executions
Thomas Kunz, Michiel F. H. Seuren
November 1997 **Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborativ**
Full text available: pdf(4.21 MB) Additional Information: full citation, abstract, references, index terms

Understanding distributed applications is a tedious and difficult task. Visualizations based on process-time dia
to obtain a better understanding of the execution of the application. The visualization tool we use is Poet, an e
at the University of Waterloo. However, these diagrams are often very complex and do not provide the user w
overview of the application. In our experience, such tools display repeated occurrences of non-trivial commun

6 ABLE: A LISP-based layout modeling language with user-definable procedural models for storage/logic
Gary B. Goates, Suhas S. Patil
June 1981 **Proceedings of the 18th conference on Design automation**
Full text available: pdf(692.32 KB) Additional Information: full citation, abstract, references, index terms

ABLE, an array-based linguistic editor, is a layout modeling language for storage/logic arrays (SLA's) that is b
programming language. This paper describes ABLE's design, presents an ABLE layout program, and evaluates
SLA-based circuit design. ABLE embodies a linguistic approach to computer-aided design (CAD) for very large
(VLSI) circuits; digital system designers can represent SLA-based integrated circuits as relatively abstract and

7 The family of concurrent logic programming languages
Ehud Shapiro
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Full text available: pdf(9.62 MB) Additional Information: full citation, abstract, references, citings, index terms

Concurrent logic languages are high-level programming languages for parallel and distributed systems that of
both known and novel concurrent programming techniques. Being logic programming languages, they preserv
the abstract logic programming model, including the logical reading of programs and computations, the conve
data structures with logical terms and manipulating them using unification, and the amenability to metaprogr

8 System-level power optimization: techniques and tools
Luca Benini, Giovanni de Micheli
April 2000 **ACM Transactions on Design Automation of Electronic Systems (TODAES)**, Volume 5 Issue 2
Full text available: pdf(385.22 KB) Additional Information: full citation, abstract, references, citings, index terms

This tutorial surveys design methods for energy-efficient system-level design. We consider electronic sytems
hardware platform and software layers. We consider the three major constituents of hardware that consume e
computation, communication, and storage units, and we review methods of reducing their energy consumptio
models for analyzing the energy cost of software, and methods for energy-efficient software design and comp

9 CASDAL: CASSM's DAta Language
Stanley Y. W. Su, Ahmed Emam
March 1978 **ACM Transacti ns n Database Systems (TODS)**, Volume 3 Issue 1
Full text available: pdf(2.72 MB) Additional Information: full citation, abstract, references, citings, index terms

CASDAL is a high level data language designed and implemented for the database machine CASSM. The langu
manipulation and maintenance of a database using an unnormalized (hierarchically structured) relational data
facilities to define, modify, and maintain the data model definition. The uniqueness of CASDAL lies in its powe
operations in terms of several new language constructs and its concepts of tagging or marking tuples and of m

Keywords: associative memory, database, nonprocedural language, query language, relational model

10 Architecture of the IBM system/370
Richard P. Case, Andris Padegs
January 1978 **C mmunications f the ACM**, Volume 21 Issue 1
Full text available: pdf(2.78 MB) Additional Information: full citation, abstract, references, citings, index terms

This paper discusses the design considerations for the architectural extensions that distinguish System/370 fr
comments on some experiences with the original objectives for System/360 and on the efforts to achieve the
reasons and objectives for extending the architecture. It covers virtual storage, program control, data-manipu
timing facilities, multiprocessing, debugging and monitoring, error handling, and input/output operations. ...

Keywords: architecture, computer systems, error handling, instruction sets, virtual storage

11 Draft Proposed: American National Standard—Graphical Kernel System
Technical Committee X3H3 - Computer Graphics
February 1984 **ACM SIGGRAPH Computer Graphics**, Volume 18 Issue SI
Full text available: pdf(16.07 MB) Additional Information: full citation

12 On randomization in sequential and distributed algorithms
Rajiv Gupta, Scott A. Smolka, Shaji Bhaskar
March 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 1
Full text available: pdf(8.01 MB) Additional Information: full citation, abstract, references, citings, index terms

Probabilistic, or randomized, algorithms are fast becoming as commonplace as conventional deterministic algo
presents five techniques that have been widely used in the design of randomized algorithms. These technique
12 randomized algorithms—both sequential and distributed— that span a wide range of applications, including
classical problem in number theory), interactive probabilistic proof s ...

Keywords: Byzantine agreement, CSP, analysis of algorithms, computational complexity, dining philosophers
algorithms, graph isomorphism, hashing, interactive probabilistic proof systems, leader election, message rou
neighbors problem, perfect hashing, primality testing, probabilistic techniques, randomized or probabilistic alg
quicksort, sequential algorithms, transitive tournaments, universal hashing

13 A language-based approach to protocol implementation
Mark B. Abbott, Larry L. Peterson
February 1993 **IEEE/ACM Transactions on Networking (TON)**, Volume 1 Issue 1
Full text available: pdf(1.88 MB) Additional Information: full citation, references, citings, index terms, review

14 A Video Compression Case Study on a Reconfigurable VLIW Architecture
D. Rizzo, O. Colavin
March 2002 **Proceedings of the conference on Design, automation and test in Europe**
Full text available: pdf(347.52 KB) Publisher Site Additional Information: full citation, abstract

In this paper, we investigate the benefits of a flexible,application-specific instruction set by adding a run-time
Functional Unit (RFU) to a VLIWprocessor. Preliminary results on the motion estimationstage in an MPEG4 vid
presented. Withthe RFU modeled at functional level and under realisticassumptions on execution latency, tech
andreconfiguration penalty, we explore different RFUinstructions at fine-grain (instruction-level) and coarse-g

15
Experience with a software-defined machine architecture

David W. Wall
May 1992 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 14 Issue 3
Full text available: pdf(2.86 MB) Additional Information: full citation, abstract, references, citings, index terms, re

We have built a system in which the compiler back end and the linker work together to present an abstract m considerably higher level than the actual machine. The intermediate language translated by the back end is th all high-level compilers and is also the only assembly language generally available. This lets us do intermodul which would be harder if some of the code in the program had come from a traditional assembler, out of sigh

Keywords: RISC, graph coloring, intermediate language, interprocedural, optimization, pipeline scheduling, p allocation, register windows

16 Code optimization - I: Optimizing memory accesses for spatial computation
Mihai Budiu, Seth C. Goldstein
March 2003 **Proceedings of the international symposium on Code generation and optimization: feed runtime optimization**
Full text available: pdf(1.06 MB) Publisher Site Additional Information: full citation, abstract, references, index terms

In this paper we present the internal representation and optimizations used by the CASH compiler for improvi parallelism of pointer-based programs. CASH uses an SSA-based representation for memory, which compactl control-flow-and dependence information.In CASH, memory optimization is a four-step process: (1)first an in representation of memory dependences is built; (2) next, unnecessary memory dependences are removed us ...

17 Programming languages for distributed computing systems
Henri E. Bal, Jennifer G. Steiner, Andrew S. Tanenbaum
September 1989 **ACM Computing Surveys (CSUR)**, Volume 21 Issue 3
Full text available: pdf(6.50 MB) Additional Information: full citation, abstract, references, citings, index terms, re

When distributed systems first appeared, they were programmed in traditional sequential languages, usually few library procedures for sending and receiving messages. As distributed applications became more common sophisticated, this ad hoc approach became less satisfactory. Researchers all over the world began designing languages specifically for implementing distributed applications. These languages and their history, their unde

18 Measurement and evaluation of the MIPS architecture and processor
Thomas R. Gross, John L. Hennessy, Steven A. Przybylski, Christopher Rowen
August 1988 **ACM Transactions on Computer Systems (TOCS)**, Volume 6 Issue 3
Full text available: pdf(2.30 MB) Additional Information: full citation, abstract, references, citings, index terms, re

MIPS is a 32-bit processor architecture that has been implemented as an nMOS VLSI chip. The instruction set based. Close coupling with compilers and efficient use of the instruction set by compiled programs were goals The MIPS architecture requires that the software implement some constraints in the design that are normally hardware implementation. This paper presents experimental results on the effectiveness of this processor ...

19 A language-based approach to protocol implementation
Mark B. Abbott, Larry L. Peterson
October 1992 **ACM SIGCOMM Computer Communication Review , Conference proceedings on Commun architectures & protocols**, Volume 22 Issue 4
Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, references, citings, index terms

Morpheus is special-purpose programming language that facilitates the efficient implementation of communic Protocols are divided into three categories, called shapes, so that they can inherit code and data structures ba the programmer implements a particular protocol by refining the inherited structure. Morpheus optimization t layer overhead on time-critical operations to a few assembler instructions even though the ...

20 Practical data breakpoints: design and implementation
Robert Wahbe, Steven Lucco, Susan L. Graham
June 1993 **ACM SIGPLAN Notices , Pr ceedings of the ACM SIGPLAN 1993 conference on Programm**

design and implementation, Volume 28 Issue 6
Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

A data breakpoint associates debugging actions with programmer-specified conditions on the memory state o
program. Data breakpoints provide a means for discovering program bugs that are tedious or impossible to is
breakpoints alone. In practice, programmers rarely use data breakpoints, because they are either unimpleme
slow in available debugging software. In this paper, we present the design and implementation of a practical

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next